Preview

- Real-Time CPU Scheduling
  - Minimizing Latency
  - Preemptive Priority-Based Scheduling
  - Rate-Monotonic Scheduling
  - Earliest-Deadline-First Scheduling
  - Proportional Share Scheduling
- Criteria for selecting an algorithm

Real-Time CPU Scheduling

- CPU scheduling for real-time operating systems involves special issues. In general, we can distinguish between hard real-time systems and soft real-time systems.
- Hard real-time systems - A task must be serviced by its deadline; service after the deadline has expired is the same as no service at all. Any missed deadline to be a system failure.
  - An Inkjet printer has a print head with control software for depositing the correct amount of ink onto a specific part of the paper. If a deadline is missed then the print job is ruined.
  - Air France Flight 447 crashed into the ocean after a sensor malfunction caused a series of system errors.

Real-Time CPU Scheduling

- Soft real-time system allows for frequently missed deadlines, and as long as tasks are timely executed their results continue to have value. Completed tasks may have increasing value up to the deadline and decreasing value past it.
  - Weather stations have many sensors for reading temperature, humidity, wind speed, etc. The readings should be taken and transmitted at regular intervals, however the sensors are not synchronized. Even though a sensor reading may be early or late compared with the others it can still be relevant as long as it is close enough.
  - The sound system in computer. If you miss a few bits, no big deal, but miss too many and you're going to eventually degrade the system.

Real-Time CPU Scheduling
(Minimizing Latency)

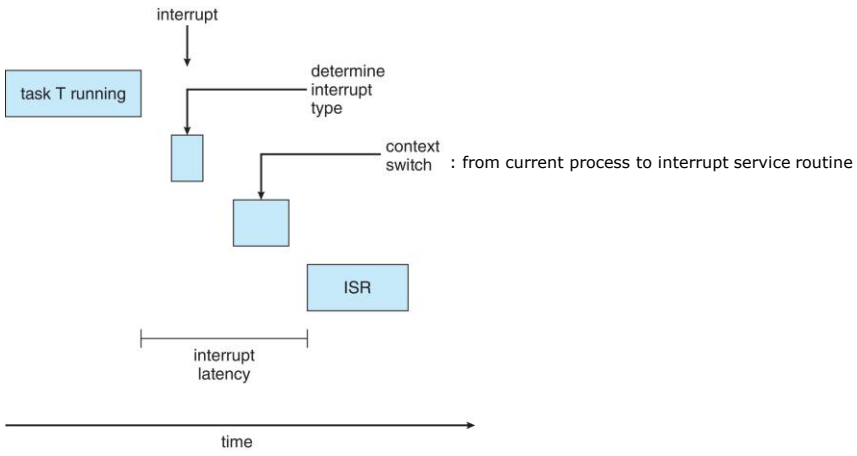
- A real-time system is typically waiting for an event in real time to occur. Events may arise either in software (as when a timer expires)or in hardware (when a remote-controlled vehicle detects that it is approaching an obstruction).
- When an event occurs, the system must respond to and service it as quickly as possible.
- **Event latency** is the amount of time that elapses from when an event occurs to when it is serviced.
- Different events have different latency requirements in a system.
  - For an antilock brake system – 3 to 5 milliseconds.
  - If antilock brake system does not respond within, car accident might occurs

Real-Time CPU Scheduling
(Minimizing Latency)

- Two types of latencies affect the performance of real-time systems

1. Interrupt latency –.
   - When an interrupt occurs,
     - the OS complete the instruction it is executing and
     - determine the type of interrupt that occurred.
     - save the state of the current process before servicing the interrupt using the specific interrupt service routine (ISR).
   - A Interrupt latency is the period of time from the arrival of an interrupt at the CPU to the start of the routine that services the interrupt
2. Dispatch latency – The amount of time required for the scheduling dispatcher to stop one process and start another

Real-Time CPU Scheduling
(Minimizing Latency)



Real-Time CPU Scheduling
(Minimizing Latency)

- Two types of latencies affect the performance of real-time systems

1. Interrupt latency –
2. Dispatch latency –
   - Dispatch latency is the amount of time required for the scheduling dispatcher to stop one process and start another
   - The most effective technique for keeping dispatch latency low is to provide preemptive kernels. For hard real-time systems, dispatch latency is typically measured in several microseconds.
   - Two components of conflict phase of dispatch latency:
     - Preemption of any process running in the kernel
     - Release by low-priority processes of resources needed by a high-priority process

Real-Time CPU Scheduling
(Minimizing Latency)

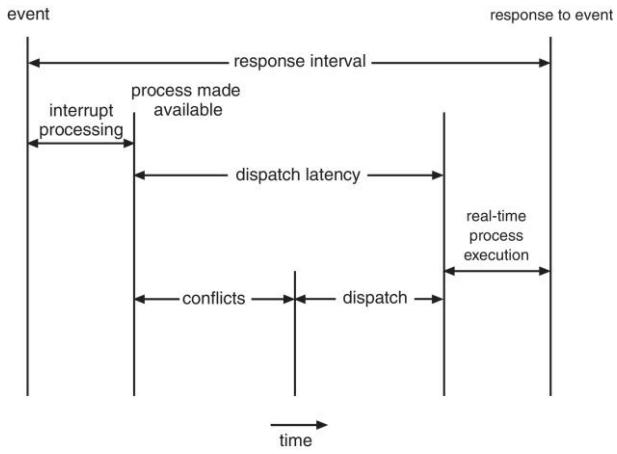


Fig. Dispatch Latency

Real-Time CPU Scheduling
(Preemptive Priority-Based Scheduling)

- The most important feature of a real-time operating system is to respond immediately to a real-time process.
- In Priority-Based scheduling, scheduler always select highest priority process; more important tasks are assigned higher priorities than those deemed less important.
- With Preemptive, process currently running on the CPU will be preempted if a higher-priority process becomes available to run.
- Preemptive, priority-based scheduler only guarantees soft real-time functionality
- Since hard real-time system need consider deadline, the scheduler requires additional scheduling features.

Real-Time CPU Scheduling

- Characteristics for a process require to consider for real-time CPU scheduler.
  - Period (p)- That is, process require the CPU at constant intervals (periods).
  - Deadline (d)
  - Processing time (t)–fixed processing time
- The relationship of the processing time, the deadline, and the period can be expressed as
  - $0 \leq t \leq d \leq p$.
- The rate of a periodic task is 1/p.

Real-Time CPU Scheduling

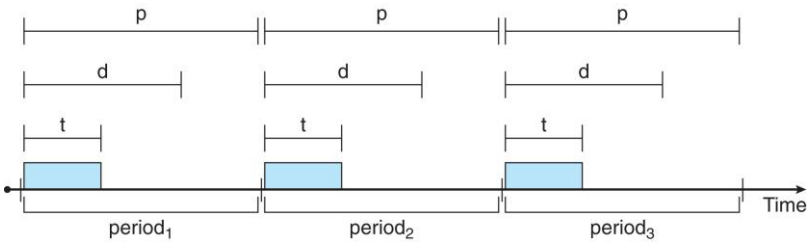


Fig. Periodic task.

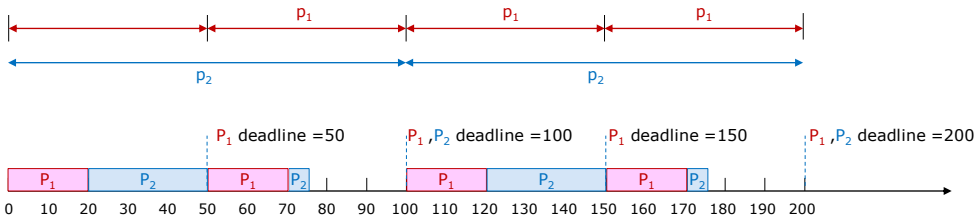
- A process have to provide its deadline requirement to the scheduler. Then, the scheduler either admit the process guaranteeing that the process will complete on time, or rejects the request as impossible (admission-control algorithm)

Real-Time CPU Scheduling
(Rate-Monotonic Scheduling)

- Rate-monotonic scheduling assumes that the processing time of a periodic process is the same for each CPU burst. That is, every time a process acquires the CPU, the duration of its CPU burst is the same.
- Upon entering the system, priority is assigned to a process based on length of period (p). The shorter the period, the higher the priority.
- The rate-monotonic scheduling algorithm schedules periodic tasks using a static priority policy with preemption.
- If a lower-priority process is running and a higher-priority process becomes available to run, it will preempt the lower-priority process.

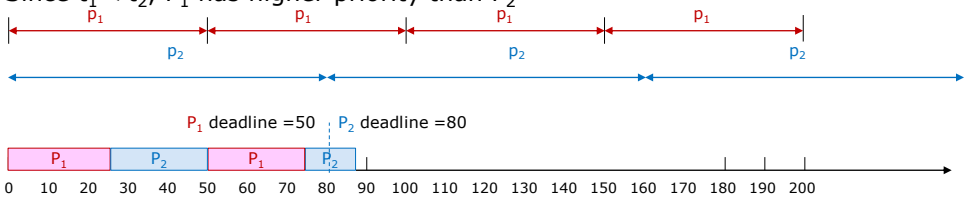
Real-Time CPU Scheduling
(Rate-Monotonic Scheduling)

- Ex) Two process P_1, P_2
  - p_1= *50*, t_1=*20*, d_1= by the start of its next period
    - CPU utilization of P_1= p_1/t_1=20/50 = 0.4
  - p_2=*100*, t_2=*35*, d_2 = by the start of its next period
    - CPU utilization of P*1*= p_2/t_2=35/100 = 0.35
  - Since $t_1 < t_2$, P_1 has higher priority than P_2



Real-Time CPU Scheduling
(Rate-Monotonic Scheduling)

- Ex) Two process P_1, P_2
  - p_1= *50*, t_1=*25*, d_1= by the start of its next period
    - CPU utilization of P_1= p_1/t_1=25/50 = 0.5
  - p_2=*80*, t_2=*35*, d_2 = by the start of its next period
    - CPU utilization of P_2 = p_2/t_2=35/80 = 0.4375
  - Since $t_1 < t_2$, P_1 has higher priority than P_2



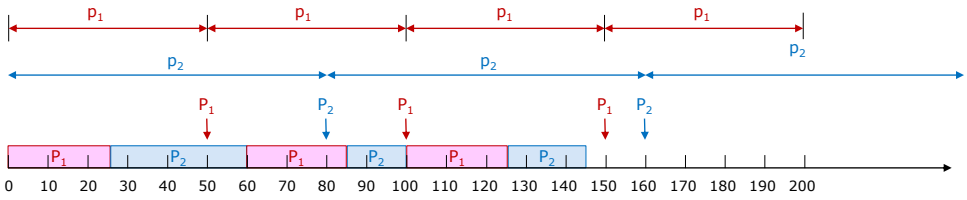
  - rate-monotonic scheduling cannot guarantee that they can be scheduled so that they meet their deadlines.

Real-Time CPU Scheduling
(Earliest-Deadline-First Scheduling)

- Earliest-deadline-firs (EDF) scheduling assigns priorities dynamically according to deadline.
  - The earlier the deadline, the higher the priority;
  - The later the deadline, the lower the priority
- When a process becomes runnable (ready state), it must announce its deadline requirements to the system.
- Priorities have to be adjusted to reflect the deadline of the newly runnable process.

Real-Time CPU Scheduling
(Earliest-Deadline-First Scheduling)

- Ex) Two process P_1, P_2
  - p_1= *50,* t_1=*25,* d_1= by the start of its next period
    - CPU utilization of P_1= p_1/t_1=25/50 = 0.5
  - p_2=*80,* t_2=*35,* d_2 = by the start of its next period
    - CPU utilization of P_2 = p_2/t_2=35/80 = 0.4375



Real-Time CPU Scheduling
(Earliest-Deadline-First Scheduling)

- EDF scheduling is theoretically optimal—theoretically, it can schedule processes so that each process can meet its deadline requirements and CPU utilization will be 100 percent.
- In practice, it is not possible to achieve this level of CPU utilization due to the cost of context switching between processes and interrupt handling.

Real-Time CPU Scheduling
(Proportional Share Scheduling)

- Proportional share schedulers operate by allocating T shares among all processes.
- An process can receive N shares of time, ensuring that the process will have N/T of the total processor time.
- Ex) There are three processes P_1, P_2, and P_3. And, total of share T = 100.
  - P_1 is assigned 50 shares, P_2 is assigned 15 shares, and P_3 is assigned 20 shares.
  - Means that P_1 will have 50 %, P_2 will have 15 %, and P_3 will have 20 % of total processor time.

Real-Time CPU Scheduling
(Proportional Share Scheduling)

- Proportional share schedulers must work in conjunction with an admission-control policy to guarantee that an application receives its allocated shares of time.
- An admission-control policy will admit a process requesting a particular number of shares only if sufficient shares are available.
  - In previous example, we have allocated 50 + 15 + 20 = 85 shares of the total of 100 shares.
  - If a new process P_4 requested 30 shares, the admission controller would deny P_4 entry into the system.

Scheduling Algorithm Evaluation

- Criteria for selecting an algorithm.
  - CPU Utilization
  - Throughput
  - Response time